Верификация программно-конфигурируемых сетей при помощи системы UPPAAL

Владислав Подымов, Ульяна Попеско

МГУ имени М.В. Ломоносова
valdus@yandex.ru, ulya_kiber@mail.ru

Аннотация В последние несколько лет активное развитие получили программно-конфигурируемые сети (ПКС) – особый вид компьютерных сетей, в которых все коммутирующие устройства имеют централизованное управление. В статье исследуются задачи формального описания и верификации ПКС. Для описания ПКС используется библиотека элементов UML в редакторе диаграмм Dia. Для верификации ПКС используется программно-инструментальное средство UPPAAL. Основной результат исследований - разработка транслятора, позволяющего по диаграмме сети получить её модель для верификации в виде сети конечных временных автоматов. Корректность алгоритма трансляции строго обоснована. Проведен ряд экспериментов, которые показывают применимость предложенного метода верификации для проверки свойств поведения ПКС, специфицированных посредством формул темпоральной логики реального времени.

Keywords: программно-конфигурируемая сеть, верификация, временные автоматы, темпоральная логика, UPPAAL

1 Введение

Идея программно-конфигурируемых сетей (ПКС) сформулирована специалистами университетов Стэнфорда и Беркли в 2006 году [1]. В таких сетях уровень управления отделен от устройств передачи данных: коммутаторы не участвуют в определении маршрутов для пакетов, а только реализуют программу контроллера. Наиболее широко применяемым стандартом для построения ПКС является протокол OpenFlow [2].

Сеть OpenFlow состоит из коммутаторов, управляемых централизованным контроллером. Пакет, передаваемый по сети, обрабатывается контроллером существенно медленнее, нежели коммутатором, поэтому одной из основных функций контроллера является организация работы коммутаторов так, чтобы они обрабатывали большую часть пакетов, и лишь в исключительных случаях пакеты обрабатывались бы на контроллере.

Организация работы коммутаторов заключается в установке правил в таблицы коммутации (flow tables), определяющих, как будут обрабатываться те или иные пакеты. Правило состоит из шаблона, идентифицирующего вид пакетов, целочисленного приоритета, устраняющего неоднозначность в

случае наложения шаблонов, целочисленного лимита времени, указывающего число секунд до истечения срока активности правила, и списка действий, описывающих обработку пакета. Также для каждого правила в этих таблицах коммутатор содержит счетчики для учета количества и размера обрабатываемых пакетов.

Коммутатор обрабатывает приходящий пакет в 3 этапа. Сначала он выбирает правило из своей таблицы коммутации так, чтобы заголовок пакета соответствовал шаблону этого правила. Если такового не найдено, коммутатор отправляет пакет контроллеру для дальнейшей обработки. Иначе выбирается совпадающее по шаблону правило с наибольшим приоритетом. На втором шаге обновляются счетчики для выбранного правила. И наконец, к пакету поочередно применяются все действия, записанные в правиле. К допустимым действиям относятся $output(op)$ и $modify(h,n)$. По действию $output(op)$ пакет отправляется на порт op. Действие $modify(h,n)$ предписывает переписать заголовочное поле h на n.

Контроллер устанавливает правила в таблицы коммутации, реагируя на события в сети. Такими событиями являются подключение нового коммутатора к сети, удаление ранее действующего коммутатора из сети, события для сбора статистики, истечение лимита времени правила коммутатора. Также контроллер оперирует функциями отправки сообщений коммутаторам: установкой правил в таблицу коммутации, удалением всех правил с заданным шаблоном из таблицы коммутации, отправкой пакета и действия для его обработки коммутатором.

В статье исследуется возможность верификации ПКС как распределенных систем реального времени. Для этого потребовалось решить следующие задачи: 1) выбрать адекватное средство для построения сетей; 2) выбрать подходящее средство для верификации сетей как систем реального времени; 3) построить корректный транслятор из выбранного средства построения сетей во входной язык нужной системы верификации; 4) провести экспериментальное исследование возможности применения выбранного средства верификации для проверки спецификаций ПКС.

Стандарт OpenFlow включает в себя большое количество свойств и предписаний для коммутации пакетов в сети. Производители компонентов компьютерных сетей используют лишь часть из них. Мы также ограничимся в рассматриваемых вариантах. При моделировании правил таблиц коммутации не будем уделять внимание приоритетам и счетчикам, а из всех допустимых действий будем рассматривать только действия типа $output(op)$. Сбор статистических данных в сети также не будет нас интересовать. С другой стороны, в нашу модель будут добавлены временные характеристики, описывающие работу физических объектов сети. Посредством этого мы будем учитывать не только предписания стандарта OpenFlow, но и технические возможности коммутаторов и каналов.

2 Схема верификации

Для построения ПКС был выбран кроссплатформенный редактор диаграмм Dia[1]. Сеть из рассматриваемого нами подкласса описывается с помощью объектов библиотеки элементов UML, предоставляемых редактором, следующим образом (см. рисунок 1). Каждый коммутатор изображается в виде прямоугольника с тремя секциями. В верхней секции обозначается имя коммутатора, в нижней содержатся правила таблицы коммутации, в средней — физические характеристики коммутатора. Каждое правило таблицы коммутации имеет четыре поля: имя порта, на который поступил пакет, заголовок поступившего пакета, лимит времени жизни правила и порт, на который необходимо отправить пакет. Описываются следующие характеристики коммутатора: *port* — множество портов коммутатора, соединяющих его с другими коммутаторами и внешней средой; *rule_imp* — задержка поиска и выполнения правила на коммутаторе (временной интервал); *rule_def* — задержка установки в коммутатор правила от контроллера; *rule_ar* — интенсивность поступления пакетов из внешней среды; *rule_con* — задержка доставки необработанного пакета контроллеру; *tab* — объем таблицы коммутации, то есть максимальное число правил таблицы.

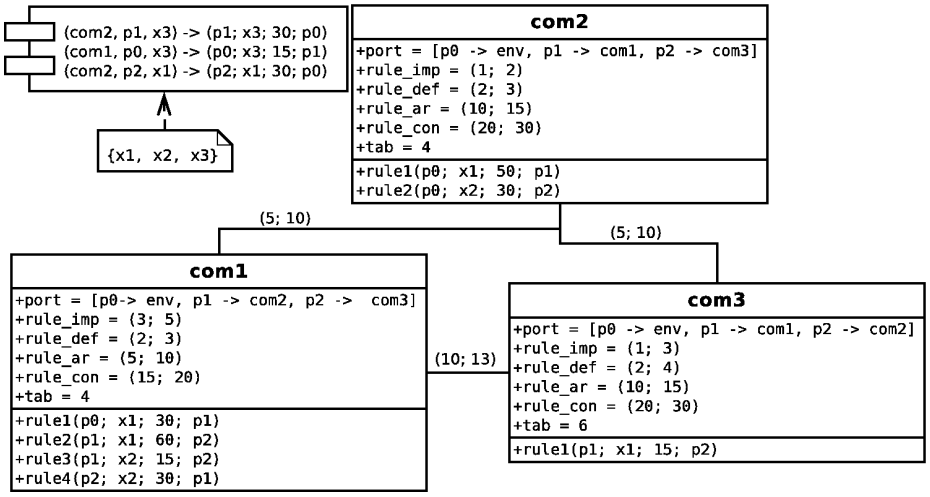

Рис. 1. Пример описания ПКС диаграммами Dia.

Для отображения топологии сети коммутаторы соединяются линиями, моделирующими дуплексные каналы сети. Каналы также имеют временную характеристику — задержку доставки пакета. Поведение контроллера опре-

[1] Руководство пользователя редактора Dia доступно по адресу http://dia-installer.de/doc/en/dia-manual.pdf

деляется политикой коммутации пакетов в сети, которая для нашего класса задач представима в виде таблицы, отображающей соответствие между заголовком поступившего на контроллер пакета и правилом, которое необходимо установить на коммутатор, приславший этот пакет. Также отдельно описывается множество всех возможных заголовков пакетов.

Для проверки спецификаций построенных таким образом ПКС было выбрано программно-инструментальное средство UPPAAL [3], использующее метод проверки свойств на моделях. Данный метод предполагает наличие модели, описывающей систему на определенном уровне абстракции, и позволяет проверить, удовлетворяет ли заданная модель системы формальным спецификациям. В средстве верификации UPPAAL в качестве модели используются сети конечных временных автоматов (параллельные композиции временных автоматов) [4], а формальные спецификации задаются формулами темпоральной логики TCTL [3].

Временные автоматы представляют собой конечные автоматы, работающие в реальном времени и осуществляющие синхронизацию посредством передачи сигналов через каналы связи. Особенностью таких автоматов является возможность использования таймеров. Значения таймеров можно указывать во временных ограничениях условий переходов между состояниями. Показания всех таймеров изменяются на одинаковые величины с течением времени.

Для верификации ПКС, описанных в терминах диаграмм Dia, средством UPPAAL, нами предложен алгоритм трансляции диаграмм в сети временных автоматов. Сеть, получаемая при трансляции, состоит из автоматов, моделирующих коммутаторы, контроллер, внешнюю среду и каналы сети. Таким образом, в результате трансляции диаграмм мы получаем сеть, пригодную для верификации средством UPPAAL. Подобный подход к верификации распределенных систем реального времени был применен в работе [5].

3 Формальный синтаксис ПКС

Перед описанием алгоритма трансляции необходимо ввести формальную модель ПКС. С учетом выбранных нами ограничений ПКС может быть описана системой (H, con, Com, $Chan$), где H — множество заголовков пакетов сети, con — контроллер, $Com = (com_1, \ldots, com_n)$ — набор коммутаторов сети и $Chan = (c_1, \ldots, c_m)$ — набор каналов сети. Заметим, что во введенной формальной модели ПКС вместо пакетов рассматриваются только их заголовки, при этом содержащиеся в пакетах сообщения опускаются. Для простоты восприятия далее под пакетами в формальной модели будут подразумеваться их заголовки.

Коммутатор com_i включает в себя множество портов $Port_i$, начальную таблицу коммутации $Rule_i \subseteq Port_i \times H \times Real \times Port_i$ объема tab и временные характеристики L_i^{imp}, R_i^{imp}, L_i^{def}, R_i^{def}, L_i^{ar}, R_i^{ar}, L_i^{con}, R_i^{con}, естественным образом соотносящиеся с характеристиками коммутатора, описанными

в диаграмме (см., например, рисунок 1). Правило (p, h, x, p') таблицы коммутации означает, что пакет h, пришедший на порт p, перенаправляется на порт p'. При этом x — время жизни правила; если $x \geq 0$, то правило назовем активным, иначе истекшим. Шаблоном правила (p, h, x, p') будем называть пару (p, h).

Контроллер *con* представляет собой множество пар вида (i, r), означающих, что правило r может быть выслано коммутатору com_i. Канал c описывается тем, из какого порта какого коммутатора он исходит, в какой порт какого коммутатора он входит и временными характеристиками L, R — минимальное и максимальное время задержки при доставке пакета. Заметим, что в предложенной модели каналы являются однонаправленными. Такое ограничение несущественно, т.к. дуплексный канал моделируется двумя однонаправленными.

4 Формальная семантика ПКС

Для доказательства корректности алгоритма трансляции необходимо ввести формальное описание работы ПКС. Данное описание является формализацией поведения ПКС в рамках стандарта OpenFlow с учетом введенных нами ограничений.

Состояние ПКС складывается из состояний контроллера и всех коммутаторов и каналов. Для удобства описания предполагаем, что каждый коммутатор com_i располагает следующими каналами: канал c_i^{ar} входного потока с временными характеристиками $L = L_i^{ar}$, $R = R_i^{ar}$, ведущий в специально выделенный под него порт; канал $c_i^{to\,con}$, ведущий в контроллер из другого специально выделенного порта, с характеристиками $L = L_i^{con}$, $R = R_i^{con}$; канал $c_i^{from\,con}$, ведущий от контроллера в третий специально выделенный порт коммутатора, с характеристиками $L = R = 0$. Таким образом, состояние S ПКС включает в себя состояния s^{con} контроллера, $s_1^{com}, \ldots, s_n^{com}$ коммутаторов и $s_1^{ch}, \ldots, s_m^{ch}$, $s_1^{ar}, \ldots, s_n^{ar}$, $s_1^{tocon}, \ldots, s_n^{tocon}$, $s_1^{fromcon}, \ldots, s_n^{fromcon}$ каналов *ch* между коммутаторами, *ar* от входного потока, $to\,con$ к и $from\,con$ от контроллера соответственно.

Коммутатор com_i может находиться в состояниях $(start, Rule)$, $(select, t, Rule, h, p)$, $(hit, Rule, h, p)$, $(miss, Rule, h, p)$, $(mod, t, Rule)$. В состоянии *start* коммутатор прослушивает входящие каналы на предмет поступивших пакетов. В состоянии *select* коммутатор, получив на порт p пакет h, просматривает таблицу коммутации для принятия решения о перенаправлении пакета. В состоянии *hit* пакет засылается в канал, исходящий из порта p. В состоянии *miss* шаблон, состоящий из пакета h и порта p, на который он пришел, засылается в канал к контроллеру. Вообще говоря, коммутатор также должен посылать контроллеру свой идентификатор, однако в построенной формальной модели идентификатор может быть восстановлен по номеру порта, входящего в коммутатор. В состоянии *mod* происходит запись в таблицу коммутации правила, присланного контроллером. Во всех состояниях

коммутатора $Rule$ — текущая таблица коммутации, t — время нахождения в текущем управляющем состоянии, h и p — хранимые пакет и порт.

Каналы могут могут находиться в состояниях $(empty)$, $(full, t, o)$ и $(sent, o)$, где o — пакет для обычных и поточных каналов, шаблон для каналов к контроллеру либо правило для каналов от контроллера. Компонента $t \in Real$ — время обработки сообщения каналом.

Контроллер может находиться только в состояниях $(idle)$ (ожидание запросов от коммутаторов) и $(send, i, r)$ (послать i-му коммутатору новое правило r).

Начальное состояние S_0 системы строится из начальных состояний коммутаторов, контроллера и каналов. Начальное состояние i-го коммутатора — $(start, Rule_i)$, контроллера — $(idle)$, каналов — $(empty)$.

Работа ПКС характеризуется последовательностью состояний, начинающейся в S_0 и строящейся по описанным далее правилам. Запись $s \to s'$ означает, что следующее состояние может быть получено из предыдущего заменой компоненты s на s'. Запись $s_1, s_2 \to s_1', s_2'$ означает то же самое для пары компонент. Построение вычисления ПКС происходит по следующим правилам.

1. Получение пакета: s_i^{com}, $s_i^{ar} = (start, Rule)$, $(sent, h) \to (select, 0, Rule, h, p)$, $(empty)$.
2. Применение активного правила $r = (p, h, x, p')$: $s_i^{com} = (select, Rule, t, h, p) \to (hit, Rule, h, p')$, если $t \in [L_i^{imp}, R_i^{imp}]$, $r \in Rule$.
3. Обработка истекшего правила $r = (p, h, x, p')$: $s_i^{com} = (select, Rule, t, h, p) \to (miss, Rule', h, p)$, если $t \in [L_i^{imp}, R_i^{imp}]$, $r \in Rule$ и $Rule' = Rule \setminus \{r\}$.
4. Сброс пакета: $s_i^{com} = (select, Rule, t, h, p) \to (start, Rule)$, если $t \in [L_i^{imp}, R_i^{imp}]$, $|Rule| = tab_i$, все правила из $Rule$ активны и ни одно из них не имеет шаблона (p, h).
5. Несоответствие шаблонов: $s_i^{com} = (select, Rule, t, h, p) \to (miss, Rule', h, p)$, где $Rule'$ получается из $Rule$ удалением всех истекших правил.
6. Начало перезаписи таблицы: s_i^{com}, $s_i^{from\,con} = (start, Rule)$, $(sent, rule) \to (mod, 0, Rule)$, $(sent, rule)$.
7. Успешная перезапись таблицы: s_i^{com}, $s_i^{from\,con} = (mod, t, Rule)$, $(sent, (p, h, x, p')) \to (hit, Rule', h, p')$, $(empty)$, если $t \in [L_i^{def}, R_i^{def}]$, $|Rule| < tab_i$ и $Rule' = Rule \cup \{(p, h, x, p')\}$.
8. Неуспешная перезапись таблицы: s_i^{com}, $s_i^{from\,con} = (mod, t, Rule)$, $(sent, rule) \to (start, Rule)$, $(empty)$, если $t \in [L_i^{def}, R_i^{def}]$ и $|Rule| = tab$.
9. Пересылка пакета коммутатору: s_i^{com}, $s_j^{ch} = (hit, Rule, h, p)$, $(empty) \to (start, Rule)$, $(full, 0, h)$, если канал c_j исходит из порта p коммутатора i.
10. Пересылка шаблона контроллеру: s_i^{com}, $s_i^{to\,con} = (miss, Rule, h, p)$, $(empty) \to (start, Rule)$, $(full, 0, (p, h))$.
11. Успешная обработка шаблона контроллером: s^{con}, $s_i^{to\,con} = (idle)$, $(sent, (p, h)) \to (send, i, r)$, $(empty)$, если $(i, (p, h, x, p')) \in con$.
12. Неуспешная обработка шаблона контроллером: s^{con}, $s_i^{to\,con} = (idle)$, $(sent, (p, h)) \to (idle)$, $(empty)$, если $(i, (p, h, x, p')) \notin con$.

13. Пересылка правила от контроллера: s^{con}, $s_i^{from\,con} = (send, i, r)$, $(empty) \to (idle)$, $(full, 0, r)$.
14. Появление пакета в сети: $s_i^{ar} = (empty) \to (full, 0, h)$, если $h \in H$.
15. Сброс пакета в окружение: $s_i^{com} = (hit, Rule, h, p) \to (start, Rule)$, если ни один канал c_j не исходит из порта p коммутатора com_i.
16. Доставка в канале: $(full, t, o) \to (sent, o)$, если $t \in [L, R]$ для характеристик L, R соответствующего канала.
17. Продвижение времени: таймеры всех коммутаторов и каналов увеличиваются на одну и ту же положительную величину d, времена жизни правил уменьшаются на эту же величину, и при этом:
    - если какой-либо канал находится в состоянии $(full, t, o)$, то $t+d \le R$ для характеристики R этого канала;
    - если $s_i^{com} = (select, Rule, t, h, p)$, то $t + d \le R_i^{imp}$;
    - если $s_i^{com} = (mod, Rule, t)$, то $t + d \le R_i^{def}$.

5 Алгоритм трансляции

Алгоритм Alg переводит ПКС в эквивалентную ей сеть временных автоматов. Понятие эквивалентности обсуждается в следующем разделе.

Сеть N, получаемая в результате трансляции ПКС $((com_1, \ldots, com_n), con, (c_1, \ldots, c_m), H)$, содержит автоматы $Hurry$, Env, $Stream$, $Chan$, A_{con} и A_i, $i \in \{1, \ldots, n\}$.

Каждый канал c_i моделируется булевыми переменными $full[c_i]$ (пакет послан), $ready[c_i]$ (пакет доставлен), таймером $t[c_i]$ и переменными для хранения объектов, пересылаемых через канал. Сброс пакетов в окружение моделируется с помощью особого канала c_{env}. Также в сеть добавляются все каналы c_i^{ar}, $c_i^{to\,con}$, $c_i^{from\,con}$.

Автомат $Hurry$ обеспечивает срочные дуги (т.е. дуги, выполняющиеся немедленно по выполнении их предусловий) и состоит из одной вершины и петли, принимающей сигнал по срочному каналу $hurry$. К срочным дугам по умолчанию добавляется посылка сигнала по каналу $hurry$. Автомат Env удаляет пакеты из канала c_{env} и состоит из одной вершины и срочной петли с записью в переменную $full[c_{env}]$ значения $false$. Автомат $Stream$ генерирует пакеты, состоит из одной вершины и содержит набор срочных петель, недетерминированно засылающий пакеты в каналы c_i^{ar}. Автомат $Chan$ обеспечивает доставку пакетов каналами, состоит из одной вершины и содержит петлю для каждого канала c, помеченную предусловием $full[c]\,\&\&\,!ready[c]\,\&\&\,(t[c] \ge L(c))$ и записью в переменную $ready[c]$ значения $true$, где $L(c)$ — характеристика L канала c. Сама вершина автомата $Chan$ при этом помечена инвариантом — конъюнкцией неравенств $t[c] \le R(c)$.

Автомат A_{con} имеет два обычных состояния — $idle$ и $send$ — и одно срочное состояние l. Срочная дуга $idle \to l$ записывает в локальные переменные автомата шаблон, присланный коммутатором, и номер этого коммутатора и в зависимости от наличия отправляемого обратно правила переходит либо

обратно в *idle*, либо в *send*. Срочная дуга $send \to idle$ присваивает переменной $full[c]$ значение $true$ и записывает в переменную канала выбранное правило.

Автомат A_i моделирует работу коммутатора com_i и состоит из обычных состояний *start*, *select*, *hit*, *miss*, *mod*, соединенных через срочные вершины. Срочная дуга $start \to select$ записывает в локальные переменные автомата доставленный через канал c пакет и порт, на который он пришел, и записывает в переменную $full[c]$ значение $false$. Состояние *select*, помеченное инвариантом $t \leq R_i^{imp}$, имеет одну исходяющую дугу в срочное состояние l, помеченную предусловием $t \geq L_i^{imp}$. Здесь t — локальный таймер коммутатора. Из состояния l в зависимости от наличия шаблона происходит переход в состояния *hit* и *miss* с модификацией таблицы согласно правилам 2-5 семантики ПКС. Удаление многих правил из таблицы коммутации обеспечивается срочной вершиной l' с петлей, удаляющей из таблицы истекшие правила, и исходящей дугой, помеченной предусловием, утверждающим, что из таблицы удалены все истекшие правила. Срочная дуга $start \to mod$ записывает в локальные переменные автомата правило, доставленное каналом $c_i^{from\,con}$. Состояние *mod* помечено инвариантом $t \leq R_i^{def}$, исходящие из него дуги — предусловием $r \geq L_i^{def}$. В зависимости от того, заполнена ли таблица коммутации, из состояния *mod* автомат может либо перейти в состояние *start*, либо перейти в состояние *hit* с записью правила в таблицу.

6 Корректность алгоритма трансляции

Под корректностью алгоритма *Alg* понимается равновыполнимость формул, используемых в средстве UPPAAL, для исходной ПКС N и результирующей сети $Alg(N)$. Ключевым понятием в доказательстве корректности алгоритма *Alg* является эквивалентность по прореживанию (stuttering equivalence) для систем переходов [6]. Неформально, такая эквивалентность означает, что если в каждом вычислении двух данных систем склеить все одинаковые состояния, то мы получим одинаковые множества вычислений. Одинаковыми полагаются состояния с совпадающими множествами выполнимых формул. В работе [7] сформулирована следующая теорема.

Теорема 1 *Если системы переходов M_1, M_2 эквивалентны по прореживанию и формула Φ логики LTL_{-X} истинна для M_1, то она также истинна для M_2.*

Следующая теорема позволяет применить только что сформулированную к системам переходов, описывающим поведение ПКС N и сети $Alg(N)$. Система переходов, описывающая поведение сети временных автоматов, обсуждается, например, в [4]. Пусть TS_A — система переходов, описывающая поведение системы A.

Теорема 2 *Пусть N — произвольная ПКС. Тогда системы переходов TS_N и $TS_{Alg(N)}$ эквивалентны по прореживанию.*

Теорему обосновывают следующие рассуждения. Состояниям контроллера и коммутаторов ставятся в соответствие одноименные состояния получающихся из них автоматов вместе с необходимыми значениями локальных переменных. Состояниям каналов ставятся в соответствие наборы значений переменных $full$, $ready$ и переменных для хранения данных. Одному применению семантических правил 1-17 соответствует последовательность переходов в сети временных автоматов через срочные состояния, и смена значений переменных происходит только в одном месте последовательности. В результате применения правила в ПКС и построенной по нему последовательности переходов в сети соответствующие состояния переводятся в соответствующие.

Корректность алгоритма напрямую следует из приведенных теорем с учетом того, что формулы, проверяемые средством UPPAAL, могут быть переформулированы в терминах логики LTL_{-X}.

7 Экспериментальное исследование

В приложении приведена сеть автоматов, построенная транслятором по описанию сети на рисунке 1. Ниже приведены проверенные с помощью средства UPPAAL свойства и их запись на языке запросов UPPAAL [3].

1. В работе системы не возникает блокировки:

$$A[]\ not\ deadlock$$

2. В сеть всегда будут поступать пакеты из внешней среды:

$$A <>\ \ forall(num : int[0, 2])\ (channel\_h[stream.align[num]])$$

3. Допустим сценарий работы сети, в котором коммутатор не принимает ни одного пакета:

$$E[]\ com1.start$$

4. При любом сценарии работы сети контроллер обработает хотя бы один пакет:

$$E <> !con.idle$$

5. Хотя бы один пакет будет успешно перенаправлен коммутатором (т.е. коммутатор выполняет свою главную функцию):

$$E <>\ \ com1.hit$$

Свойства 1,2,4,5 соответствуют спецификации сетей OpenFlow, в то время как свойство 3 является недопустимым. Проверка показала, что свойства 1,2,4,5 выполняются, а свойство 3 не выполняется. Это свидетельствует о том, что функционирование полученной сети временных автоматов соответствует нашим ожиданиям и что предложенная схема верификации ПКС

с помощью средства UPPAAL позволяет проверять свойства поведения ПКС как системы реального времени.

В ячейках таблицы 1 представлено время проверки описанных темпоральных свойств для некоторых моделей ПКС: 1 – три коммутатора в топологии кольца (рисунок 1); 2 – четыре коммутатора в топологии звезды; 3 – четыре коммутатора в произвольной топологии; 4 – два коммутатора с изначально пустыми таблицами коммутации. Проверка проводилась на вычислительном устройстве со следующими характеристиками: INTEL Core i7 3820/1600 МГц/2Гб DDR3. Первое свойство не было проверено на моделях 1–3 в связи с нехваткой памяти.

Таблица 1. Время проверки свойств ПКС.

	Свойство номер:				
	1	2	3	4	5
Модель 1	-	1 секунда	1 секунда	7 секунд	1 секунда
Модель 2	-	1 секунда	1 секунда	1 минута 2 секунды	1 минута 25 секунд
Модель 3	-	1 секунда	1 секунда	1 минута	1 минута 19 секунд
Модель 4	27 часов	1 секунда	1 секунда	1 секунда	1 секунда

8 Заключение

В результате проведенных исследований мы подтвердили практическую осуществимость подхода, предложенного в статье [5], для проверки выполнимости свойств поведения моделей программно-конфигурируемых сетей в реальном времени. Предложенное нами средство анализа поведения ПКС может быть использовано для наглядного описания конфигурации и коммутационной политики сети в виде диаграмм. Разработанный нами алгоритм трансляции преобразует диаграмму в сеть временных автоматов, подаваемую на вход системы верификации UPPAAL. Корректность алгоритма трансляции строго обоснована, транслятор реализован на языке программирования Perl. Получаемая на выходе транслятора сеть временных автоматов позволяет проверять спецификации ПКС с помощью системы UPPAAL. Особенностью такой верификации является возможность учитывать временной аспект поведения ПКС как распределённых систем реального времени.

Список литературы

1. Casado M., Garfinkel T., Akella A., Fredman M., Boneh D., McKeown N., Shenker S. SANE: A Protection Architecture for Enterprise Networks // 15-th Usenix Security Symposium, Vancouver, Canada, August 2006.

2. McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J. Openflow: Enabling innovation in campus network // SIGCOMM Computer Communication Review, 2008, v.38, n.2, p.69-74.
3. Behrmann G., David A., Larsen K. A tutorial on Uppaal // Lecture Notes in Computer Science, 2004, v.3185, p.200-236.
4. Alur R., Dill D. Automata for modeling real-time systems // Proc. of Int. Colloquium on Algorithms, Languages, and Programming, LNCS, 1990, v.443, p.322-335.
5. Волканов Д.Ю., Захаров В.А., Зорин Д.А., Коннов И.В., Подымов В.В. Как разработать простое средство верификации систем реального времени // Моделирование и анализ информационных систем, 2012, Том 19, №6, с.45–56.
6. Browne M.C., Clarke E.M., Grumberg O. Characterizing finite Kripke structures in propositional temporal logics // Theor. Comp. Sci., 1988, v.59(1-2), p.115-131.
7. Clarke E.M., Grumberg O., Peled D. Model Checking. The MIT Press. 1999.

Приложение

Сеть временных автоматов, полученная в результате трансляции из ПКС, изображенной на рисунке 1, приведена на рисунках 2–4. В целях удобочитаемости выражения автоматов написаны в синтаксисе, отличном от синтаксиса UPPAAL и при этом более простом для восприятия.

Запись вида $(i = 1..3, 5, 7)$ означает перебор всех i из заданного диапазона и создание копий дуги для каждого значения i. Запись вида $(i = 1..3 \to \Phi)$ означает "для всех i из заданного диапазона верна формула Φ".

Функция $get(c)$ сохраняет содержимое канала c в локальные переменные (h, p, ...) и выполняет присваивание $c = false$. Функция $send(c, o)$ копирует o в содержимое канала и выполняет присваивания $c = true$, $c\_ready = false$, $c\_t = 0$. Функция $set\_rule(i)$ копирует локальные переменные коммутатора, отвечающие компонентам правила, в i-ю позицию таблицы.

Предикат $to\_deliver(c)$ описывается как $c \,\&\&\, !c\_ready \,\&\&\, (c\_t \geq c\_L)$. Предикат $ok(c)$ описывается как $!c \,||\, c\_ready \,||\, (c\_t \leq c\_R)$.

Канал $c[0]$ выделен под окружение, каналы $c[1]$, $c[2]$, $c[3]$ — под потоки, прикрепленные к соответствующим коммутаторам. Автоматы A_2, A_3 отличаются от автомата A_1 только номерами задействованных каналов и константами, определяемыми количеством портов и объемом таблицы, и по этой причие опущены.

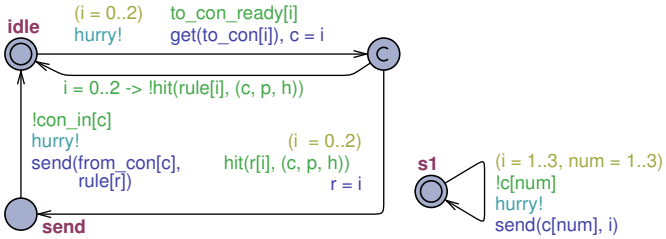

Рис. 2. Автоматы A_{con} (слева), $Stream$ (справа).

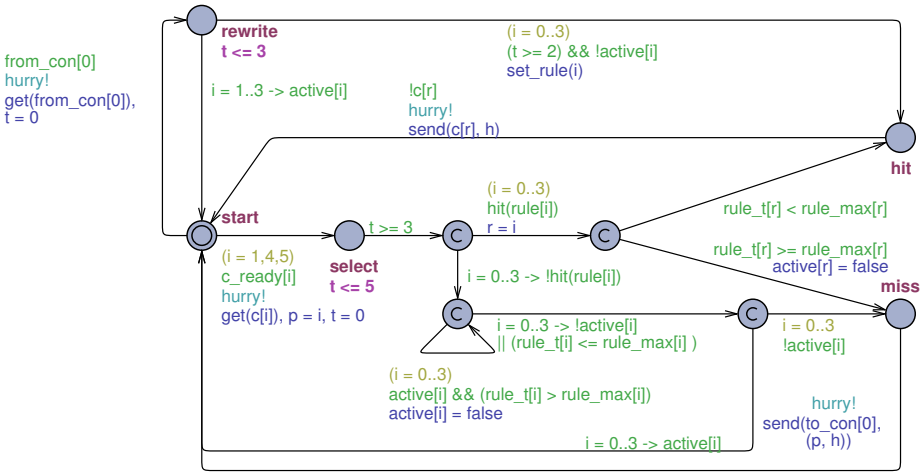

Рис. 3. Автомат A_1.

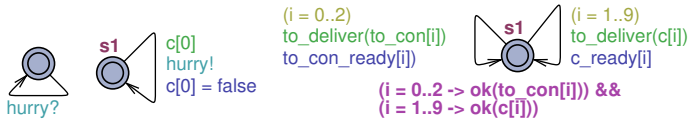

Рис. 4. Автоматы $Hurry$ (слева), Env (в центре), $Chan$ (справа).